Tecnologie del Linguaggio Naturale - Parte I
Prof. Alessandro Mazzei

Mario Bifulco

A.A. 2022/2023

Capitolo 1

Consegna

1.1 Problema affrontato

Tra le opzioni presentate come progetti finali del corso ho scelto di svolgere la prima traccia, ovvero implementare l'algoritmo CKY[1] e applicarlo su una grammatica per la lingua *Klingon*.

Il progetto è diviso in tre sotto punti fondanti:

1. Realizzare l'algoritmo senza supporto di librerie esterne
2. Testare la bontà di quanto realizzato con la grammatica L1 presente nel testo di Dan Jurafsky [2], immagini 17.10 e 17.8, con alcune frasi d'esempio
3. Scrivere la grammatica per la lingua *Klingon* e testare l'algoritmo su di essa

[1] Cocke-Kasami-Younger

[2] https://web.stanford.edu/ jurafsky/slp3/17.pdf

Capitolo 2

Algoritmo e strutture dati

2.1 Implementazione dell'algoritmo

2.1.1 Analisi dello pseudocodice

L'algoritmo CKY rappresenta una tecnica di *parsing bottom-up*, utilizzata per l'analisi sintattica di frasi basate su una grammatica *context-free*. L'algoritmo utilizza una tabella di analisi dinamica, che viene popolata con le combinazioni di regole grammaticali che possono generare le sottostringhe della frase in esame. La tabella viene quindi utilizzata per determinare se la frase di input può essere generata dalla grammatica in questione, e per costruire l'albero sintattico associato.

L'algoritmo prende come input la lista di parole che compongono la frase e una grammatica *context-free* in CNF[1]. Dopo aver costruito la tabella basta controllare la cella corrispondente all'angolo in alto a destra, se tale cella contiene il token di start la frase è corretta, ovvero è producibile con le regole espressa dalla grammatica. Vedi algoritmo 1.

Algorithm 1 CKY-parse

Require: words
Require: G
  for $j\ From\ 1\ To\ LENGTH(words)$ **do**
    for $A|A \rightarrow words[j] \in G$ **do**
      $table[j-1, j] \leftarrow table[j-1, j] \cup A$
    for $i\ From\ j-2\ To\ 0$ **do**
      for $k\ From\ i+1\ To\ j-1$ **do**
        for $A|A \rightarrow BC \in G \wedge B \in table[i,k] \wedge C \in table[k,j]$ **do**
          $table[i, j] \leftarrow table[i, j] \cup A$
  return table

[1] Chomsky Normal Form

L'algoritmo presentato sul libro di testo e sopra citato permette di effettuare un controllo sulla correttezza grammaticale della frase.

Se invece l'obiettivo fosse visualizzare l'albero sintattico è necessario utilizzare una struttura d'appoggio in modo da memorizzare i *backpointers*.

2.1.2 Strutture dati utilizzate

L'implementazione proposta dell'algoritmo CKY utilizza una matrice di insiemi per memorizzare le informazioni riguardanti le possibili combinazioni di simboli della grammatica nella frase analizzata. In altre parole, la matrice viene utilizzata per tenere traccia di quali regole grammaticali sono state utilizzate per derivare le sottostringhe della frase. Questo consente di verificare se la frase è ben formata secondo la grammatica specificata.

Inoltre, l'algoritmo utilizza un dizionario per ricostruire i *backpointers*, cioè le informazioni necessarie per generare la derivazione della frase a partire da regole grammaticali utilizzate per la sua analisi. In particolare, il dizionario tiene traccia delle regole prese in esame utilizzate per produrre ciascuna sottostringa nella matrice, consentendo di ricostruire la struttura sintattica della frase analizzata.

Si noti che, tramite il funzionamento dei cicli all'interno dell'algoritmo, non viene mai utilizzata la totalità della matrice, ma solo l'equivalente matrice triangolare inferiore, ovvero la diagonale e le celle sopra.

A seguire una rappresentazione generica delle strutture dati, per compattezza ci si riferisce a NT come un generico simbolo non terminale della grammatica e con T ad un generico simbolo terminale.

Matrice:

$$\begin{array}{ccc} NT^1, \ldots, NT^{N_1} & \cdots & \ldots, \text{S}, \ldots \\ \vdots & \ddots & \vdots \\ \{\} & \cdots & NT^1, \ldots, NT^{N_m} \end{array}$$

Struttura dei *backpointers*:

$$\begin{array}{rcl} (Idx_1, Idx_2, NT) & : & [T], \\ (Idx_3, Idx_4, NT) & : & [(NT, (Idx_5, Idx_6, NT)), \cdots] \\ \vdots & & \vdots \end{array}$$

Capitolo 3

Grammatiche

3.1 Grammatica per la lingua inglese

L'algoritmo CKY prevede di lavorare con grammatiche *context-free*, ossia L1, in CNF, cioè con sole proiezioni binarie o terminali. La grammatica per la lingua inglese proposta sul testo è riportata di seguito:

$S ::= NP\ VP$
$S ::= X1\ VP$
$X1 ::= Aux\ NP$
$S ::= book|include|prefer$
$S ::= Verb\ NP$
$S ::= X2\ PP$
$S ::= Verb\ PP$
$S ::= VP\ PP$
$NP ::= I|she|me$
$NP ::= TWA|houston$
$NP ::= Det\ Nominal$
$Nominal ::= book|flight$
$Nominal ::= meal|money$
$Nominal ::= Nominal\ Noun$
$Nominal ::= Nominal\ PP$
$VP ::= book|include|prefer$
$VP ::= Verb\ NP$
$VP ::= X2\ PP$
$X2 ::= Verb\ NP$
$VP ::= Verb\ PP$
$VP ::= VP\ PP$
$PP ::= Preposition\ NP$
$Det ::= that|this|the|a$
$Noun ::= book|flight$
$Noun ::= meal|money$
$Verb ::= book|include|prefer$
$Pronoun ::= I|she|me$
$Proper - Noun ::= houston|NWA$
$Aux ::= does$
$Preposition ::= from|to|on$
$Preposition ::= near|through$

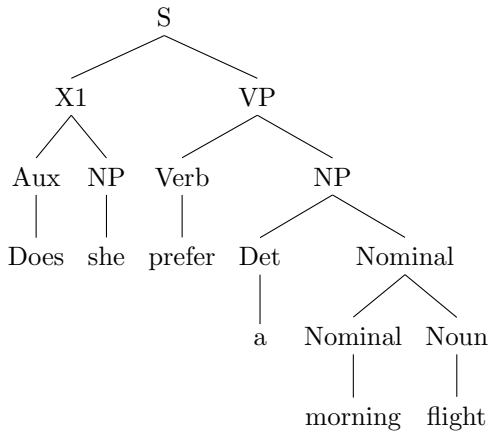
Questa grammatica è stata testata sulle seguenti frasi:

1. Does she prefer a morning flight
2. Book the flight through Houston

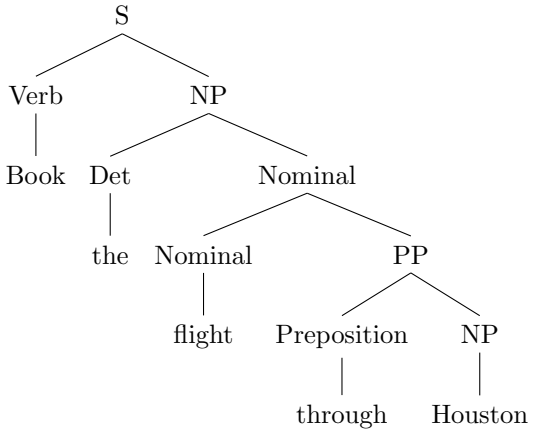
3.1.1 Risultati ottenuti

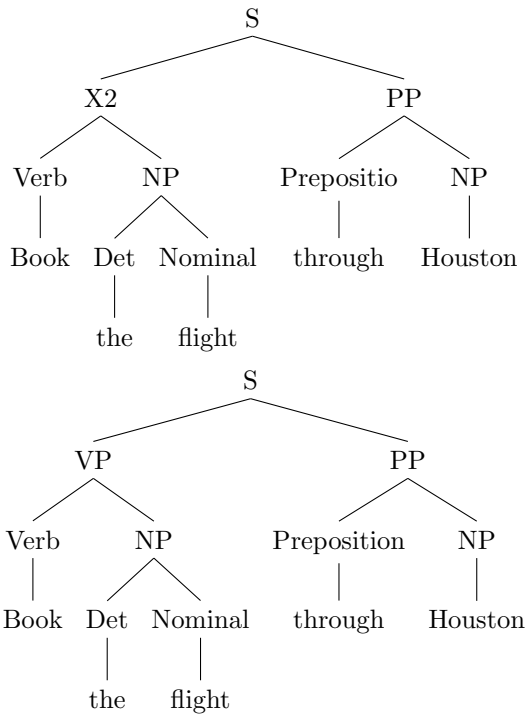
La prima non può essere prodotta con la grammatica fornita dal libro di testo in quanto manca tra le produzioni $Nominal ::= morning$.

Dopo aver ampliato la grammatica è quindi possibile eseguire l'algoritmo e analizzare la struttura dati contenente i *backpointers* per ricostruire l'albero sintattico valido.



Analizzando la seconda frase, l'algoritmo restituisce tre alberi sintattici, riportati di seguito.





Si noti che, analizzando gli alberi prodotti, è possibile trovare dei simboli non terminali che non hanno significato intrinseco nel linguaggio naturale. Ciò accade perché la grammatica è stata convertita nella forma normale teorizzata dallo studioso Chomsky, e rende necessario l'aggiunta di simboli per trasformare le produzioni ternarie in produzioni binarie.

3.2 Grammatica *Klingon*

Le lingue OVS, *Object-Verb-Subject*, rappresentano una minoranza di quelle parlate nel mondo, circa l'1%, tra queste è presente il linguaggio artificiale *Klingon*.

Per costruire gli alberi sintattici sono stati utilizzati i seguenti marcatori che identificano parti specifiche del testo: S, NP, VP, NOMINAL, ADV, PRON, VERB.

Da uno studio approfondito della lingua *Klingon*, si evidenzia che essa è molto ricca e ben documentata, ad esempio i suffissi forniscono informazioni sulla funzione dei sostantivi nella frase[1]. Tuttavia una rappresentazione semplificata della lingua permette di non considerare queste informazioni aggiuntive senza perdere rappresentatività nella visualizzazione dell'albero sintattico.

[1] https://hol.kag.org/page/Type_5:_Syntactic_markers.html

3.2.1 Regole utilizzate

Utilizzando le informazioni proposte nella pagina di *Wikipedia* relativa alla lingua *Klingon*[2] è possibile ricavare le seguenti regole linguistiche che la nostra grammatica *context-free* dovrà rispettare per poter produrre frasi accettabili:

1. Gli avverbi si trovano solitamente a inizio frase
2. Esistono tre classi di sostantivi. Tuttavia per questo progetto ci si limita ad un generico sostantivo senza distinguerne le casistiche
3. Gli aggettivi non hanno PoS, sono invece usati verbi intransitivi

3.2.2 Grammatica prodotta

La grammatica è stata creata basandosi sui principi sopra elencati e facendo riferimento alle frasi di esempio, le quali verranno ampiamente esaminate durante l'analisi dei risultati.

Poiché le frasi di esempio presentano una struttura grammaticale semplice, la creazione di una grammatica in Forma Normale di Chomsky (CNF) è risultata essere una soluzione naturale. Pertanto, non sono richieste ulteriori modifiche alla grammatica per poterla utilizzare nell'algoritmo CKY.

Il risultato di questa operazione è riportato di seguito:

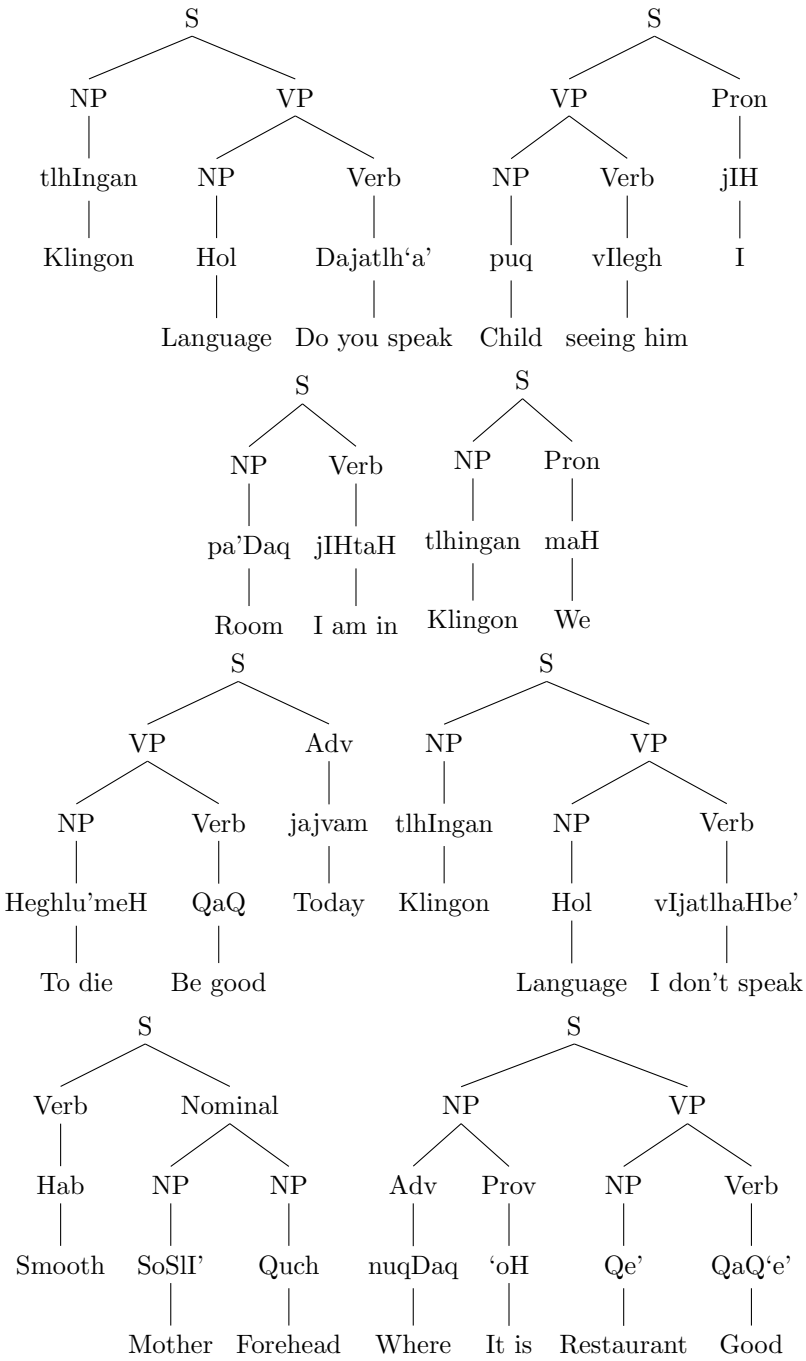
$S ::= NP\ VP$
$S ::= VP\ Pron$
$S ::= VP\ Adv$
$S ::= NP\ Pron$
$S ::= NP\ Verb$
$S ::= Verb\ Nominal$
$VP ::= NP\ Verb$
$Nominal ::= NP\ NP$
$NP ::= Adv\ Pron$
$NP ::= tlhingan|heghlu'meh$
$NP ::= hol|puq|pa'daq$
$NP ::= sosli'|quch|qe'$
$Verb ::= dajatlh'a'|vilegh$
$Verb ::= jihtah|qaq|vijatlhahbe'$
$Verb ::= hab|qaq'e'$
$Adv ::= jajvam|nuqdaq$
$Pron ::= jih|mah|'oh$

3.2.3 Risultati ottenuti

Per testare la copertura della grammatica e poter visualizzare gli alberi sintattici tramite l'algoritmo, sono state usate le frasi proposte dalla consegna arricchite con periodi disponibili online[3].

tlhIngan Hol Dajatlh'a'?
puq vIlegh jIH
pa'Daq jIHtaH
tlhIngan maH!
Heghlu'meH QaQ jajvam
tlhIngan Hol vIjatlhaHbe'
Hab SoSlI' Quch
nuqDaq 'oH Qe' QaQ'e'

[2] https://en.wikipedia.org/wiki/Klingon_grammar
[3] https://www.wikihow.it/Parlare-Klingon


S
NP
VP
tlhIngan
NP
Verb
Klingon
Hol
Dajatlh‘a’
Language
Do you speak
S
VP
Pron
NP
Verb
jIH
puq
vIlegh
I
Child
seeing him
S
NP
Verb
pa’Daq
jIHtaH
Room
I am in
S
NP
Pron
tlhingan
maH
Klingon
We
S
VP
Adv
NP
Verb
jajvam
Heghlu’meH
QaQ
Today
To die
Be good
S
NP
VP
tlhIngan
NP
Verb
Klingon
Hol
vIjatlhaHbe’
Language
I don’t speak
S
Verb
Nominal
Hab
NP
NP
Smooth
SoSlI’
Quch
Mother
Forehead
S
NP
VP
Adv
Prov
NP
Verb
nuqDaq
‘oH
Qe’
QaQ‘e’
Where
It is
Restaurant
Good

Capitolo 4

Conclusioni

La flessibilità dell'algoritmo CKY deriva dalla sua capacità di essere applicato con successo a diverse lingue, sia naturali che artificiali. Tale algoritmo si basa sulle grammatiche *context-free*, che rappresentano una descrizione formale e astratta delle regole sintattiche di una lingua. Questa proprietà consente all'algoritmo di adattarsi facilmente alle specifiche della lingua considerata, senza richiedere modifiche alla procedura implementata.

L'impiego di grammatiche *context-free* per la modellazione della sintassi delle lingue naturali e artificiali non ha evidenziato limitazioni significative per i casi esaminati. Tale formalismo è quindi consigliabile in quanto la sua complessità computazionale è contenuta ($O(n^3)$).

Tuttavia non tutte le lingue parlate sono esprimibili in questo modo e per alcune frasi è necessario utilizzare le grammatiche *mildly context sensitive*. Un esempio è il tedesco svizzero, che è composto da molte frasi con dipendenze incrociate (*Cross-serial*) per le quali non è possibile ignorare il contesto. Il formalismo delle grammatiche *mildly context sensitive* è facilmente implementabile nell'algoritmo CKY, ma questa modifica porta ad una complessità di $O(n^6)$.

L'applicabilità dell'algoritmo CKY alle diverse lingue e l'impiego di grammatiche *context-free* suggeriscono che tale tecnica di parsing possa essere utilizzata con successo in una vasta gamma di applicazioni, dalla linguistica computazionale alla programmazione.